2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

FFMRA: A Fully Fair Multi-Resource Allocation Algorithm in Cloud Environments

Hamed Hamzeh[1], Sofia Meacham[1] Kashaf Khan[2] Keith Phalp[1] and Angelos Stefanidis[1]
[1]*Faculty of Science and Technology, Bournemouth University, UK.*
hamzehh@bournemouth.ac.uk, smeacham@bournemouth.ac.uk, kphalp@bournemouth.ac.uk, astefanidis@bournemouth.ac.uk
[2]*British Telecom, Ipswich, UK, kashaf.khan@bt.com*

***Abstract*—The need for effective and fair resource allocation in cloud computing has been identified in the literature and in industrial contexts for a while. Cloud computing seen as a promising technology, offers usage-based payment, scalable and on-demand computing resources. However, during the past decade, the growing complexity of the IT world has resulted in making Quality of Service (QoS) in the cloud a challenging subject and an NP-hard problem. Specifically, the fair allocation of resources in the cloud becomes particularly interesting when many users submit several tasks which require multiple resources. Research in this area has been increasing since 2012 by introducing the Dominant Resource Fairness (DRF) algorithm as an initial attempt to solve the fair resource allocation problem in the cloud. Although DRF meets a sort of desirable fairness properties, it has been proven to be inefficient in certain conditions. Noticeably, DRF and other works in its extension are not intuitively fair after all. Those implementations have been unable to utilize all the resources in the system, leaving the system in an imbalanced situation with respect to each specific system resource. In order to address those issues, we propose in this paper a novel algorithm namely a Fully Fair Multi-Resource Allocation Algorithm in Cloud Environments (FFMRA) which allocates resources in a fully fair way considering both dominant and non-dominant shares. The results from the experiments conducted in CloudSim show that FFMRA provides approximately 100% recourse utilization, and distributing them fairly among the users while meeting desirable fairness features.**

***Keywords*-Allocation; Cloud computing; dominant; non-dominant; fairness; resource;**

I. Introduction

Cloud computing is a growing technological trend that provides on-demand, pay-as-you-use, and wide range of resources based on the Internet[1]. By its nature, it also abstracts hardware resources and simplifies the computational operations. Cloud introduces three delivery models known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Each of those layers, provisions a sort of specific services for users and providers. Due to the diversity of resources involved, cloud is a heterogeneous environment which makes resource allocation an interesting issue.

In general, tasks submitted by users require multiple resources. Some tasks are CPU intensive, like computational tasks that are mainly dependent on processing, while other tasks, such as video encoding, are RAM intensive[2]. Hence, the distribution of such resources among users remains an NP-hard problem.

There are various examples in the literature discussing resource allocation in cloud computing environments. The best ways for distributing resources among users with varying demands have been investigated extensively in[3]. As part of this work, different quality metrics were introduced and among them, fairness has been identified as an important and challenging issue , attracting more attention in recent years by the research community. Essentially, fairness is seen as an intuitive concept and one can define his/her own perspective of fairness[4]. The concepts of fairness and equality have different definitions, whereby equality is the quality of being the same quantity while fairness is the quality of having a fair situation [5].

Any fair allocation algorithm is subject to adhering to some desirable fairness features[6]. Identifying the concept of fairness is an important research question in cloud systems, which can be addressed by proposing optimization methods, suitable allocation and scheduling algorithms. How to allocate resources in a fair way is still open to investigation. As part of an early attempt, Dominant Resource Fairness (DRF) was introduced in [7],attracting much attention as a result of its good fairness features. However, it was subsequently proven to be inefficient when deployed across multiple servers, hence the use of DRF in a naive extension form leads to a highly inefficient resource allocation [8]. While previous works in this area have been limited only to dominant resources, they do not appear to be intuitively fair[9]. The unfairness problem of DRF has been recently investigated in the Apache Mesos environment [10]. However, it is useful to reconsider fairness by equalizing both dominant and non-dominant resources that could affect fairness, and the balance of the system by evenly distributing resources among the users.

In this paper, we propose FFMRA as a new fair allocation algorithm which equalizes dominant and non-dominant resources. While dominant share is the maximum share that a user has been allocated of any resource, non-dominant is the minimum amount of that resource. In FFMRA, users with dominant resource can maximize their allocation without

978-1-7281-4034-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00091

starving others.

We then show that FFMRA meets desirable fairness features. FFMRA is pareto efficient, which means that no user is able to increase her allocation without decreasing others allocations. Also, FFMRA fulfills envy-free property which means that no user prefers the allocations of other users. Additionally, FFMRA meets some other allocation features, such as sharing incentive and strategy-proof.

Finally, we analyze the performance of FFMRA in terms of resource allocation and utilization using the CloudSim simulation tool supported by randomly generated workloads. We show that FFMRA distributes resources fairly among users and provides approximately 100% utilization of resources.

The rest of the paper is organized as follows: Section II gives a brief overview of recent conducted works in DRF and its extensions. Section III illustrates the motivation behind this work. Section IV describes the proposed algorithm and related formulations. Section V represents the evaluations and experiments. Section VI analyses the fairness properties of FFMRA; and Our conclusions are drawn in section VII.

II. Related works

Fair allocation of resources has been investigated in different contexts, such as the well-known cake-cutting problem [11]. As a well-established algorithm, Max-Min fairness [12] has been commonly used in fair resource allocation in a variety of cases. Initially, most of the research in this area has been in relation to the single-resource type using Max-Min fairness, as shown by [13] [14] who assume only one type of resource, such as the CPU. However, since cloud indicates a higher variety of resources, multiple resources need to be considered. To address the inefficiency problem associated with DRF, some researchers have suggested different ways to improve its performance and efficiency in multiple server environment. Other allocation policies, as shown in [15][16][17] are not proven to be fair with respect to fairness properties like pareto-optimal and sharing incentive [7].

DRF is recognized widely as the most important fair resource allocation algorithm. In trying to address the problems associated with DRF, several attempts were made to design and develop more scalable and efficient allocation algorithms. While DRF considers only single server environments, the cloud consists of multiple servers that make DRF inefficient to use on a single server basis. In this section we review the most important research work in relation to DRF policy. The research work in [18] proposed Dominant Resource Fairness with Heterogeneous servers (DRFH) as a generalization of DRF which considers DRF on heterogeneous servers. Like DRF, DRFH guarantees desirable fairness properties. In a multi server setting, a user may have dominant resources on different servers. As such, DRFH calculates the global dominant resource of each user. The best fit heuristic design has been proposed to use the algorithm in a real world system. Although this work demonstrates improvements in terms of overall resource utilization, the results also show that RAM utilization has not reached desired utilization point. In [19] the authors proposed a new server-based algorithm to overcome the existing issues in DRF. To fulfill the trade-off between fairness and efficiency, the resource allocation was done by maximizing per-server utility functions using particular classes. Similar to other works, the algorithm in [19] meets certain good fairness features. The main point of the work conducted is to calculate the dominant resources based on each server, including the virtual dominant resource. Overall, the algorithm is shown to improve the resource utilization and meets the fairness properties. However, according to the results of some of the conducted experiments, certain resources are found not fully-utilized.

The research work in [20] suggests a fair resource allocation algorithm using Nash Bargaining (NB) mechanism and Lexicographical Max-Min Fair (LMMF) to maximize the resource utilization and meet the fair allocation properties taking into account multiple tasks submissions by users. Three metrics are considered for analyzing different mechanisms such as computational efficiency, fairness and incentive compatibility. In [21] authors propose a long-term multi-resource fairness algorithm named H-MRF. The idea behind this algorithm is that, unlike other algorithms, H-MRF targets pay-as-you-use computing systems. Since, DRF has RAM-less properties, users are able to cheat and misreport their demands, hence, H-MRF tackles this issue. The tests carried out show that H-MRF offers good performance improvements and shared benefits. In [22] a new multi resource allocation algorithm was proposed in order to solve the efficiency of fair resource allocation. DRBF takes into account the bottlenecked resources and places them in different queues, while using linear programming for resource allocation, based on dominant resources. Based on the test conducted, DRBF provides 100% resource utilization and better fair allocation. However, similar to other work in this area, it seems that DRBF is not able to provide an intuitive fair allocation, since some users with dominant resources may not able to increase their allocations. The research carried out in [23] compares DRF and Proportional Fairness (PF) in terms of efficiency and it indicates that PF is more efficient than DRF in terms of resource utilization as the resource wastage in PF is less than that of DRF. The work in [24] examines the multi type resource allocation problem in an distributed computing environment and considers Peer-to-Peer (P2P) systems which have become more popular. It also meets better fairness conditions in DRF. The results from the various tests show that this proposed algorithm outperforms DRF in terms of native extension on different servers

III. MOTIVATION

In the previous section, we explained that DRF and similar algorithms in its extension are not intuitively fair. DRF considers only dominant resources and equalizes them to calculate the allocation. This way of resource distribution may lead to an inefficient and imbalanced allocation which means that if one considers X% of CPU in the resource pool for all dominant resources, it should also consider the same X% amount of RAM. This process should also be applied for non-dominant shares to achieve a totally fair system.

At this point it is worth reviewing DRF's working scheme. DRF is the generalization of Max-Min fairness which aims to allocate resources fairly among different users competing over multiple resources. The intuition behind DRF is to equalize dominant shares of each user. As an example (see Table I), there is a system with two resource types (CPU, RAM) in which two users (A, B) submit their tasks with demand vectors (1 CPU, 4 RAM) and (3 CPU, 1 RAM) respectively. If the capacity of the resource pool is (9 CPU, 18 RAM), dominant share for user A is RAM since $(1/9 < 4/18)$ and for user B is CPU since $(3/9 > 1/18)$. Accordingly, DRF allocates (3 CPU, 12 RAM) for user A and (6 CPU, 2 RAM) for user B. So the proportion of allocation based on the total resource pool capacity is $(6/9, 12/18)$ in which $6/9 = 12/18$. So, we can say that the distribution of resources is balanced for dominant shares. If we consider non-dominant shares, the proportion is $(3/9, 2/18)$ for both users in which $(3/9 \neq 2/18)$. Hence, the allocation is not balanced for non-dominants. In that case, 4GB of RAM is left unused under DRF allocation. Lets take a look at a system with more than two users considering Dominant Resource with Bottlenecked Fairness(DRBF)[22]. DRBF is a generalization of Bottleneck Aware Allocation (BAA). It captures the concept of fairness and efficiency by dividing users in different queues with respect to dominant resources. According to table II, although DRBF provides a fair allocation and full utilization of resources, however, it is not intuitively fair so that it is not able as to distribute resource fairly among users. As can be seen in Table II, in DRBF dominant shares in CPU get totally lower proportion of resources of the resource pool compared to dominant RAM shares. Also, according to the same table, it is clear that User B with dominant share on CPU is not able to maximize her allocation that is the same for DRF, due to that the system resources are not being well-distributed, since only dominant shares are considered. To be more clear, elaborating this point further, the sum of allocations for dominant and non-dominant CPU shares in DRBF are $4.8 + 5 = 9.8$ and $3.9 + 4.6 = 8.5$ respectively. Similarly, for dominant and non-dominant RAM shares we have $15.6 + 16.1 = 31.7$ and $1.6 + 3 = 4.6$. The allocations for non-dominants are considerable high compared to the allocations for dominant shares in both resource types. In order to determine how whether the allocation is balanced or not, it is necessary to calculate the proportion of allocation for dominant shares for each specific resource type. Since, the capacities of CPU and RAM are 18 and 36 respectively, the proportion of CPU in the resource pool for all CPU dominant shares is $9.8/18 = 0.54$ and for dominant RAM shares is $31.7/36 = 0.88$ of which $0.54 \neq 0.88$. For non-dominant CPU and RAM shares, the proportions are $8.5/18 = 0.47$ and $4.6/36 = 0.12$ in which $0.47 \neq 0.12$. Interestingly, DRBF pushes most of the proportion of the resource pool to dominant RAM shares and, as a result, dominant CPU shares are not able to utilize the system resource fairly. By looking at DRF, the proportions are $11/18 = 0.61$ and $26/36 = 0.72$ for dominant CPU and RAM shares respectively in which $0.61 \neq 0.72$. For non-dominant CPU and RAM, the proportions are $7/18 = 0.38$ and $5/36 = 0.13$ respectively in which $0.38 \neq 0.13$. Similarly, under DRF allocation and according to this example, there is a considerable resource wastage in RAM since DRF utilizes only 72% of RAM in the resource pool which is not efficient. These analysis indicate that considering only dominant shares is not enough to show that a system is fair. This motivates us to propose a new fair resource allocation algorithm, namely FFMRA, in cloud computing systems. We advocate that all resources in the resource pool should be distributed evenly with respect to each resource type.

Table I: The allocation of resources in DRF and FFMRA with resource capacity (9 CPU, 18 RAM)

Users	User A	User B
Resources	(CPU , RAM)	(CPU , RAM)
Requested	(1 , 4)	(3 , 1)
DRF	(3 , 12)	(6 , 2)
FFMRA	(2 , 14)	(7 , 4)

IV. FFMRA

FFMRA is inspired by DRF, however, it considers and equalizes both dominant and non-dominant shares. In order to understand how FFMRA works, we refer to the example in previous section that two users are competing over two resources. Initially, FFMRA calculates the contribution of dominant and non-dominant resource in the resource pool. This gives a good correlation between both types of resources which helps to keep the system in a balanced condition.

A. Problem formulation

Given that d_i^k and nd_i^k represent dominant and non-dominant resources vectors of k resource types, d_i^k and nd_i^k can be calculated as follows:

$$d_i^k = max(\frac{r_i^k}{C_{max}^k}) \tag{1}$$

$$nd_i^k = min(\frac{r_i^k}{C_{max}^k}) \tag{2}$$

where r_i^k indicates requested resource type k by user i which is always positive ($r_i^k > 0$) and C_{max}^k represents the maximum capacity of resource type k. If total capacity of the resource pool(sum of the capacity of all resources) is indicated by T, then the proportion of total resources for dominant and non-dominant shares in (1) and (2) indicated by d_i^p and nd_i^p are calculated as follows:

$$d_i^p = \frac{\sum d_i^k * T}{\sum d_i^k + \sum nd_i^k} \tag{3}$$

$$nd_i^p = \frac{\sum nd_i^k * T}{\sum d_i^k + \sum nd_i^k} \tag{4}$$

Consequently, according to (3) and (4) and given that $x_1, x_2, ..., x_n$ are the number of allocated tasks for each user, in that case the allocation for each user is calculated based on the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & (x_1, x_2, ..., x_n) \\ \text{subject to} \quad & \sum_{i=1}^{n} r_i^k.x_i \leq C_i^{max}. \\ & \sum_{i=1}^{n} d_i^k.x_i \leq d_i^p. \\ & \sum_{i=1}^{n} nd_i^k.x_i \leq nd_i^p. \end{aligned} \tag{5}$$

Formulas (3) and (4) indicate the correlation between dominant and non-dominant shares since, d_i^p and nd_i^p are calculated based on the contribution of both shares in the resource pool based on $\sum d_i^k + \sum nd_i^k$.

Table II: The allocation and utilization of resources in FFMRA which is normilized by where user 1 has dominant share in RAM and user 2 has dominant share in CPU.

	CPU	RAM
User 1	0.222222	0.777778
User 2	0.777778	0.222222
utilization	1	1

Table III: The allocation and utilization of resources in DRF where user 1 has dominant share in RAM and user 2 has dominant share in CPU.

	CPU	RAM
User 1	0.333333	0.666667
User 2	0.666667	0.111111
utilization	1	0.777778

B. A scenario with two users

Based on the example in Table I, if we assume that i_1, i_2, j_1, j_2 are the number of tasks allocated to both users and the capacity of the resource pool is (9 CPU, 18 RAM), user A gets $(i_1, 4i_2)$ and user B gets$(3j_1, j_2)$. Hence, according to algorithm 1, the total demands of dominant resources are 7(four dominant RAM tasks for user A and three CPU dominant tasks for user B) which is 2 for non-dominants(one non-dominant CPU task for user A and one non dominant RAM task for user B). As far as the total number of resources in the resource pool is 27, by applying the proportionality, dominant and non-dominant shares get $21/27$ and $6/27$ of the resource pool respectively.

So, by specifying the proportion of dominant and non-dominant shares, we are ready to calculate the allocation for each user with the following optimization problems:

$$\begin{aligned} \text{maximize} \quad & (i_2, j_1) \\ \text{subject to} \quad & 4i_2 + 3j_1 \leq 21. \end{aligned}$$

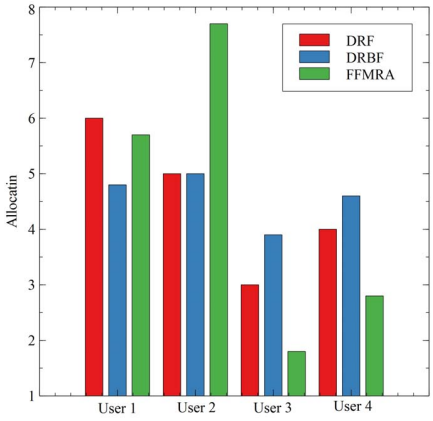
$$\begin{aligned} \text{maximize} \quad & (i_1, j_2) \\ \text{subject to} \quad & i_1 + j_2 \leq 6. \end{aligned}$$

Solving above linear optimization equations gives, $i_1 = 3.5, j_1 = 2, i_2 = 2.3$ and $j_2 = 4$. So the final allocation will be (2 CPU, 14 RAM) for user A and (7 CPU, 4 RAM) for user B. Hence, by comparing those allocations with DRF(Table I), not only the overall status of the system is fair but also the utilization is 100% for both resources. Also, according to table I, FFMRA utilizes 100% of resources and outperforms DRF in terms of resource utilization since DRF utilizes $14/18$ of RAM and leaves $4/18$ unused whereas FFMRA utilizes $18/18$ of RAM. Furthermore, DRF allocates more CPU to user A who has non-dominant share in that resource. As a result, user B with dominant CPU share gets 6 CPU whereas FFMRA allocates only 2 CPU to user A and 7 CPU for user B. At the same time, DRF is not able to offer maximum resources to user A who has dominant resource in RAM and it allocates only 12 RAM to that user, whilst FFMRA allocates 14 RAM to user A and 4 CPU for user B who has non-dominant shares in RAM. Tables II and III indicate how FFMRA maintains fairness and efficiency. Table II shows FFMRA distributing resources fairly among dominant and non-dominant shares and utilizing from the resource pool's capacity. According to table III, although DRF maintains fairness for dominant shares, however the contribution of those shares in FFMRA in the resource pool and also resource utilization is higher compared to DRF.

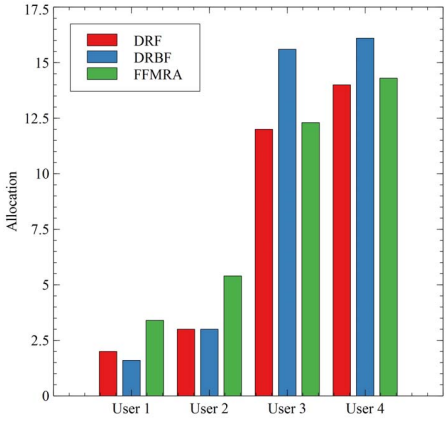
C. A scenario with more than two users

As a general solution, for a system with more users, the allocation can be calculated using algorithm 1 which gives an alternative and simple calculation to work out the allocation for each user. This is note that, after distributing

resources among all dominant and non-dominant shares of each specific resource type, the allocation for each user is relaxed to proportional sharing that can be seen in parts 13 and 14 of Algorithm 1.



(a) CPU allocation



(b) RAM allocation

Figure 1: The distribution and allocation of resources in three different algorithms.

Table IV: Resource allocation in three different algorithms with resource capacity (18 CPU, 36 RAM)

Users	User A	User B	User C	User D
Demands	3 , 1	5 , 3	1 , 5	2 , 7
DRF	6 , 2	5 , 3	3 , 12	4 , 14
DRBF	4.8 , 1.6	5 , 3	3.9 , 15.6	4.6 , 16.1
FFMRA	5.7 , 3.4	7.7 , 5.4	1.8 , 12.3	2.8 , 14.3

According to Table IV, the proportion of allocation for dominant CPU and RAM shares are $13.4/18 = 0.74$ and $26.6/36 = 0.74$ respectively of which $0.74 = 0.74$. For non-dominant CPU and RAM shares, we have $4.6/18 = 0.25$ and $8.8/36 = 0.25$ of which $0.25 = 0.25$. Therefore,

Algorithm 1 FFMRA algorithm

1: $C \leftarrow (c_1, c_2, ..., c_i)$ ▷ Capacity vector consists of capacity for each resource c_i such as CPU, RAM and etc.
2: $TC \leftarrow \Sigma c_i$ ▷ sum of the capacity of all resources
3: $U \leftarrow (u_1, u_2, ..., u_i)$ ▷ total users in the system
4: $D \leftarrow (d_1, d_2, ..., d_i)$ ▷ dominant shares vector consist of all the shares which are dominant
5: $ND \leftarrow (nd_1, nd_2, ..., nd_i)$ ▷ non-dominant shares vector consist of all the shares which are non-dominant
6: nD ▷ number of dominant shares
7: nND ▷ number of non-dominant shares
8: $SD \leftarrow \Sigma D$ ▷ sum of all dminant shares
9: $SND \leftarrow \Sigma ND$ ▷ sum of all non-dominant shares
10: $PD \leftarrow TC * SD/SD + SND$ ▷ The proportion of total resource pool capacity for dominant shares
11: $PND \leftarrow TC * SND/SD + SND$ ▷ The proportion of total resource pool capacity for non-dominant shares
12: **for** each u_i **do**
13: $A(d_i) \leftarrow d_i + (((PD * c_i)/TC)/nD)$ ▷ The allocation for each user who has dominant share in any specific resource type
14: $A(nd_i) \leftarrow nd_i + (((PND * c_i)/TC)/nND)$ ▷ The allocation for each user who has non-dominant share in any specific resource type

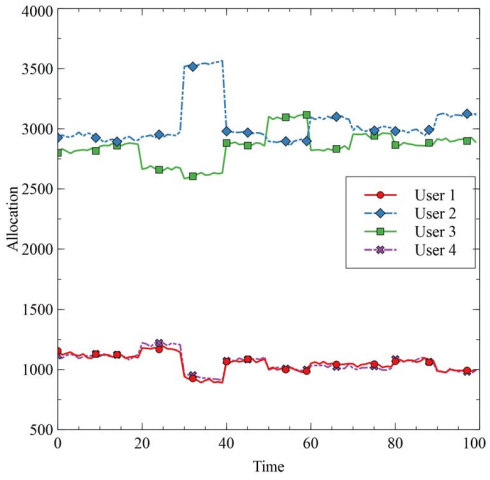
FFMRA maintains the balance in the system by distributing resources evenly among dominant and non dominant shares. Also, according to Figure 1a, FFMRA tries to allocate more resources to dominant shares so that as shown in Figure 1, DRF and DRBF allocates considerably less resources to CPU dominant shares since by looking at Figure 1b, DRF and DRBF allocates more resource to RAM dominant shares especially DRBF which considers more Ram in the resource pool to dominant shares. Considering that the allocation of CPU resource for dominant shares in DRF and DRBF has been reduced to increase RAM allocation of dominant shares, it contradicts the fair resource allocation, so the balance of the system is not maintained. However, the numerical analysis reveals the fact that FFMRA keeps the system in a balanced situation and distributes resources fairly than DRF and DRBF.

V. Performance Evaluation

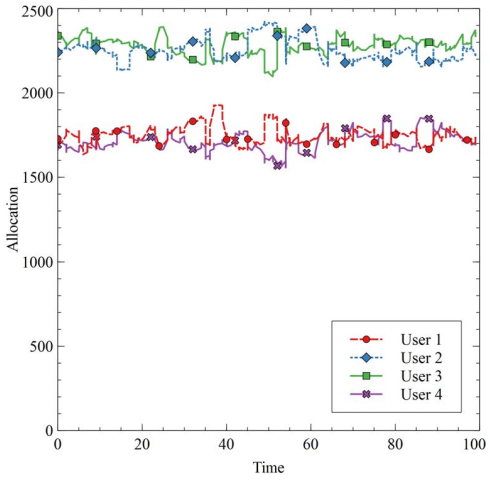
In this section we evaluate the performance of FFMRA in terms of resource allocation and utilization. We use the CloudSim simulation tool to compare FFMRA with DRF by considering four users submitted tasks over two types of resources (CPU and RAM). To compare the allocation of resources in FFMRA and DRF we setup the configuration of the system to (mips=8000, pe count=1,RAM=16384) with 100 iterations in milliseconds. All the resource are

assumed to be divisible and all the demands are positive. Workloads have been generated randomly from a stochastic data generator. In this specific experiment, users 2 and 3 are dominant in CPU and users 1 and 4 are dominant in RAM. According to Figure 2(a), FFMRA tries to allocate more resources to dominants and it considers approximately 70-75% of CPU in the resource pool for dominant shares. However, according to Figure 2(b), DRF allocates more resource to non-dominant shares. It leads to an unfair allocation since dominant shares needs more resources to run their intensive tasks. On the other hand, and according to Figure 3(a) and (b), again FFMRA allocates considerably more resources to dominant shares compared to DRF and allocates approximately 70-75% of RAM in the resource pool for dominants. So, FFMRA tries to keep the system in a stable and balanced condition. In other words, when we consider 70% of the resource pool's CPU capacity for dominant shares, it is necessary to consider the same amount of the resource pool's RAM capacity for dominants.

In order to evaluate the performance of FFMRA and compare it with DRF in terms of utilization, we conducted tests in a large-scale system using randomly selected workloads in time series simulations with 500 iterations and 300 virtual machines. According to Figures 4 and 5, FFMRA outperforms DRF in CPU and RAM utilization. Specifically, it shows extremely higher RAM utilization compared to DRF. It is worth nothing that, despite the theoretical nature of the experiments, the utilization in practical evaluations is not exactly 100% due to some users not requesting any specific resource type. We believe that the performance of FFMRA will be more tangible in the real-time environments like Apache mesos or VmWare Vsphere in which users dynamically join and depart from the system. However, at the moment, FFMRA policy gives fairer allocation and efficient utilization than DRF especially in large-scaled systems.



(a) FFMRA



(b) DRF

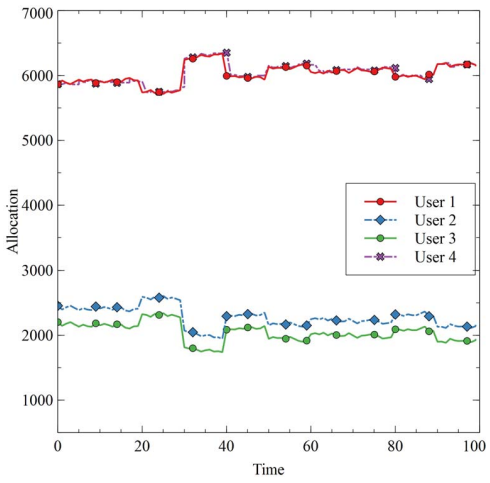
Figure 2: The allocation of CPU in DRF and FFMRA

VI. Fairness properties analysis

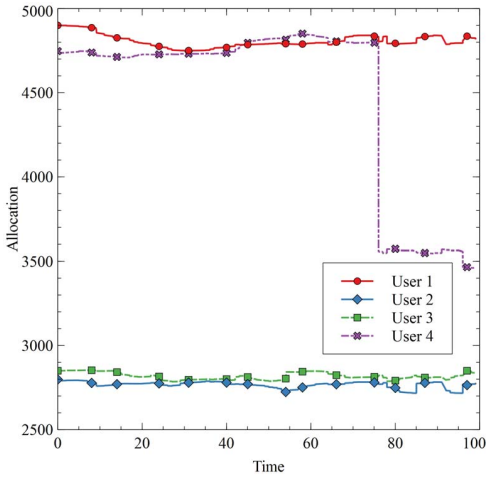
In this section we explore how FFMRA is able to meet some desirable fairness properties.

- **Theorem 1**. FFMRA satisfies envy-freeness.
  Proof: Assuming that D represents set of all dominant shares in entire system. If $R = (r_1, r_2, ..., r_i)$ indicates each specific resource in the system like CPU, RAM, and etc. based on algorithm 1 FFMRA considers equal proportion of the resource pool capacity for each specific resource in R. So, as an example, for each specific resource in R if we assume r_i and $r_i^{'}$ are two users with dominant shares so that $r_i > r_i^{'}$ and both get the allocation based on Max-Min fairness algorithm which allocates resources according to what they ask for and divides the remaining equally among the users, hence, $r_i^{'}$ is not be able to envy r_i. Therefore, FFMRA meets envy-free property.

  Proof: Given that we have two groups of tasks denoted by r, r^*, and FFMRA increases the allocation of dominant shares of all users in each group based on the maximum share by proportionality. Indeed, FFRDRF sums up all dominant shares together and gives them the highest proportion of total capacity of the system. So, by balancing the load in each specific resource, we make sure that each dominant resource share will get at least $1/n$ of resource capacity. As an example in the second scenario with four users, PD for dominant resources over RAM is 26.6. So, this is more than the half of the RAM capacity in which by applying Max-Min fairness, we guarantee that FFMRA satisfies sharing incentive property.
- **Theorem 3** FFMRA satisfies pareto-efficient.
  Proof: Again, assuming that r, r^* denote RAM and CPU intensive tasks respectively. Any resource of a task



(a) FFMRA



(b) DRF

Figure 3: The allocation of RAM in DRF and FFMRA

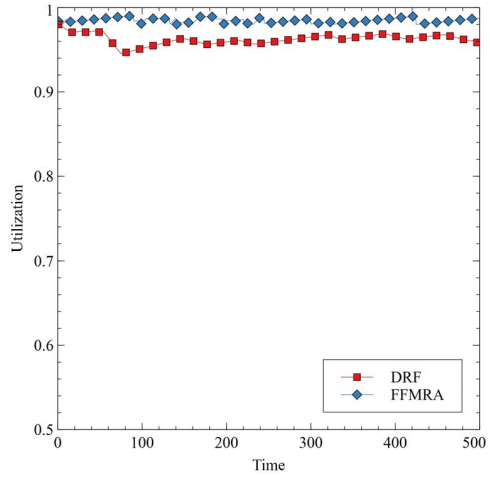


Figure 4: CPU utilization

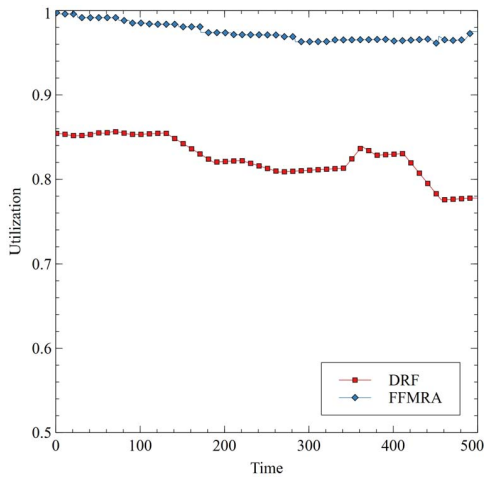


Figure 5: RAM utilization

in r, r^* is able to increase its dominant resource without decreasing the allocation of other tasks. In another word, if there are two users i and j which are using a saturated resource r, then increasing the dominant share of user i would be decreasing the dominant share of user j. However, in every step of FFMRA algorithm by increasing the dominant resource of a user, another users dominant share is increased as well. So, the algorithm is pareto-efficient.

- **Theorem 4**. FFMRA meets strategy-proof in which users are not able to misreport their demands.
  Proof: Assume that a user considers demand vectors d_r and $d_r^{'}$ in which d_r and $d_r^{'}$ denote true and misreported demands respectively. Given that FFMRA increases dominant shares based on available resources in each stage, if a user with d_r tries to manipulate the server by $d_r^{'}$ and considering that the capacity constraint is taken into account, in that case the constraint will be violated by misreporting the true demand by any user. So, it is not possible for a user to misreport her demand under FFMRA allocation policy.

VII. Conclusion

In this paper, we proposed FFMRA as a new fair allocation algorithm in cloud environments, inspired by DRF as the first fair resource allocation algorithm in the cloud. Although DRF has good fairness features, it contains certain drawbacks in terms of efficiency and fairness. We presented that considering only dominant shares is not enough to meet fairness in the cloud. Hence, by taking into account both dominant and non-dominant resources we attempted to provide a new fair allocation algorithm. In order to evaluate our proposed algorithm, we compared FFMRA with DRF. Our comparison showed that DRF is not able to maintain the system in a balanced state and some users may not

be able to increase their allocation to meet their needs. Based on the same results, FFMRA gives around 100% utilization of resources and guarantees that each user gets their desired resources. As part of the attempt to further this work, we are currently in the design and implementation stage of this algorithm which is being developed with the BT Group plc and it is based on the on their cloud application requirements. The CloudSim environment was selected for the first phase of the work to simulate and get initial results before deploying the algorithm to the cloud platform. We are currently working on applying this algorithm within the companys in-house agile cloud methodologies. Last but not least, we plan to extend our work to areas of user experience and socio-technical algorithms, and investigate the societal impact of fairness algorithms in several application contexts.

REFERENCES

[1] R. Buyya, J. Broberg, and Goscinski Andrzej, Cloud computing: principles and paradigms. Hoboken, NJ: Wiley, 2011.

[2] G. Lee, B-G. Chn and R. H. Kats,*Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud*, Proceedings of the 3rd USENIX conference on Hot topics in cloud computing. June, 2011.

[3] F. L. Pires and B. Barn, *Cloud computing resource allocation taxonomies*, International Journal of Cloud Computing, vol. 6, no. 3, p. 238, 2017.

[4] M. Fleurbaey, Defining fairness,*Fairness, Responsibility, and Welfare* , pp. 1540, 2008.

[5] M. Li and D. P. Tracer, *Interdisciplinary Perspective on Fairness, Equity, and Justice*. S.l.: Springer, 2019.

[6] E. Meskar and B. Liang, *Fair multi-resource allocation with external resource for mobile edge computing*, IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2018.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, *Dominant resource fairness: Fair allocation of multiple resource types.*inProc. USENIX NSDI , vol. 11, 2011, pp. 2424.

[8] W. Wang, B. Li, and B. Liang, *Dominant resource fairness in cloud computing systems with heterogeneous servers*, IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014.

[9] P. Poullie, T. Bocek and B. Stiller, *"A Survey of the State-of-the-Art in Fair Multi-Resource Allocations for Data Centers*," in IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 169-183, March 2018. doi: 10.1109/TNSM.2017.2743066

[10] P. Saha, A. Beltre and M. Govindaraju, "Exploring the Fairness and Resource Distribution in an Apache Mesos Environment," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2018, pp. 434-441.doi: 10.1109/CLOUD.2018.00061

[11] S. J. BRAMs and A. D. Taylor, *Fair division,* 1996.

[12] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalo *Upward Max Min Fairness*, 2012 Proceedings IEEE INFOCOM, 2012.

[13] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, *Quincy: Fair scheduling for distributed computing clusters*. In SOSP 09, 2009.

[14] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. *Delay Scheduling*.

[15] H. Moulin., *Fair Division and Collective Welfare*. The MIT Press, 2004.

[16] H. Varian., *Equity, envy, and efciency. Journal of Economic Theory*, 9(1):6391, 1974.

[17] H. P. Young., *Equity: in theory and practice*. Princeton University Press, 1994.

[18] W. Wang, B. Li, and B. Liang, *Dominant resource fairness in cloud computing systems with heterogeneous servers,* IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014.

[19] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Zhao, *An Efficient and Fair Multi-Resource Allocation Mechanism for Heterogeneous Servers,* IEEE Transactions on Parallel and Distributed Systems, pp. 11, 2018.

[20] D. Zarchy, D. Hay, and M. Schapira, Capturing resource tradeoffs in fair multi-resource allocation, 2015 IEEE Conference on Computer Communications (INFOCOM), 2015.

[21] S. Tang, Z. Niu, B. He, B.-S. Lee, and C. Yu, *Long-Term Multi-Resource Fairness for Pay-as-you Use Computing Systems,* IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 5, pp. 11471160, Jan. 2018.

[22] L. Zhao, M. Du and L. Chen, *"A new multi-resource allocation mechanism: A tradeoff between fairness and efficiency in cloud computing"*, China Communications, vol. 15, no. 3, pp. 57-77, 2018.

[23] Y. Jin and M. Hayashi, Efficiency comparison between proportional fairness and dominant resource fairness with two different type resources, 2016 Annual Conference on Information Science and Systems (CISS), 2016.

[24] Q. Zhu and J. C. Oh, An Approach to Dominant Resource Fairness in Distributed Environment, Current Approaches in Applied Artificial Intelligence Lecture Notes in Computer Science, pp. 141150, 2015.